Self-Organizing Maps for Structured Data

M. Hagenbuchner[1], A. Sperduti[2], and A-C. Tsoi[1]
[1]*Faculty of Informatics, University of Wollongong, Australia*
[2]Dipartimento di Informatica, Università di Pisa, Italy

Abstract. In this paper we are interested in discovering similarities among complex objects which can reasonably be represented by labelled acyclic directed graphs. We show that it is possible to extend the SOM model to study this type of structured inputs. We demonstrate the capabilities of the proposed model by utilizing a relatively large data set taken from an artificial benchmark problem involving visual patterns encoded as labelled DAGs.

1 Introduction

Many natural and artificial systems are more appropriately modelled using data structures. For example, an image can be segmented into various components, and each component can be considered in relation to one another in the form of a graph structure. This structured representation conveys much more information than a "flat" one, such as a vector of numerical features extracted from the image. In fact, when considering an image retrieval system, the adoption of this type of representations should allow for the formulation of much more complex queries as well as the achievement of improved retrieval results. The use of structured representations is convenient also for representing electronic documents, chemical compounds, and many other structured objects.

Processing of data structures, i.e., rooted directed acyclic graphs (DAGs) by neural networks has been considered by a number of researchers recently, notably [7]. Unfortunately, the technique introduced by Sperduti & Starita cannot be used if the supervising signal is missing.

In this paper, we propose to use Self-Organising Maps as building blocks of a coding procedure, which is similar to the one used in [7], in order to learn structural similarities in the data. Specifically, we show how Self-Organizing Maps (SOMs) [6] can be extended to process complex objects represented by data structures. Standard SOMs have been especially studied in the past for their ability to preserve the main topological relationships among feature vectors in high dimensional input vectorial spaces when projected onto a lower dimensional display space. Basically, a SOM implements a discrete approximation of principal curves [9] by means of a set of neural units arranged into a topological map.

The ability of the SOM to preserve topological features can be exploited to perform a clustering of the data on the basis of a similarity criterion and also to induce a metric onto the input domain. Here we extend the standard SOM model, so as to allow the mapping of structured objects into a topological map. This mapping can then be used to discover similarities among the input objects.

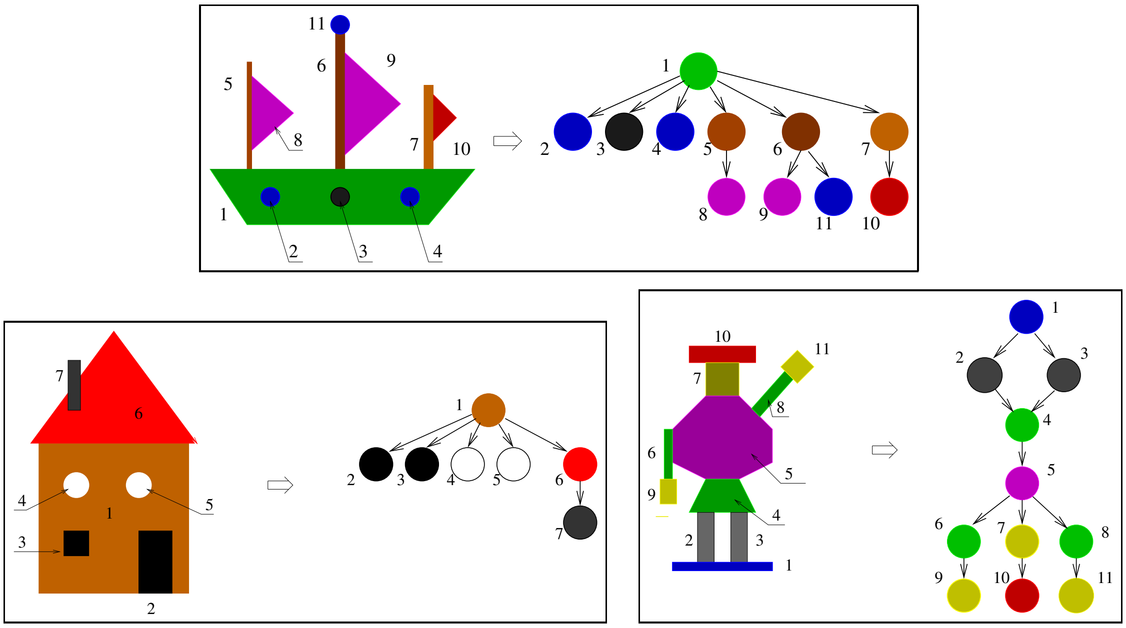

Figure 1: Examples of DOAGs representing visual patterns. Labels are not shown.

2 Data Structures and Notation

We shall use lowercase bold letters to denote vectors, uppercase bold letters to denote matrices, and calligraphic letters for representing graphs. A data structure $\mathcal{Y}$ is a DAG whose vertices are labelled by vectors of real-valued numbers which either represent numerical or categorical variables. Subscript notation will be used when referencing the labels attached to vertices in a data structure. Hence $\boldsymbol{y}_v$ denotes the vector of variables labelling vertex $v \in \text{vert}(\mathcal{Y})$.

In this paper we assume that instances in the learning domain are DOAGs (directed ordered acyclic graphs) or DPAGs (directed positional acyclic graphs). A DOAG is a DAG $\mathcal{D}$ with vertex set $\text{vert}(\mathcal{D})$ and edge set $\text{egd}(\mathcal{D})$, where for each vertex $v \in \text{vert}(D)$ a total order on the edges leaving from v is defined. DPAGs are a superclass of DOAGs in which it is assumed that for each vertex v, a bijection $P : \text{egd}(\mathcal{D}) \rightarrow I\!N$ is defined on the edges leaving from v. The *indegree* of node v is the number of incoming edges to v, whereas the *outdegree* of v is the number of outgoing edges from v.

We shall require the DAG (either DOAG or DPAG) to possess a supersource[1], i.e. a vertex $s \in \text{vert}(\mathcal{D})$ such that every vertex in $\text{vert}(\mathcal{D})$ can be reached by a directed path starting from s. Given a DAG $\mathcal{D}$ and $v \in \text{vert}(\mathcal{D})$, we denote by $\text{ch}[v]$ the set of children of v, and by $\text{ch}_k[v]$ the k-th child of v.

In the following, we shall denote by $\#^{(i,c)}$ the class of DAGs with maximum indegree i and maximum outdegree c. A generic class of DAGs with bounded (but unspecified) indegree and outdegree, will simply be denoted by $\#$. The class of all data structures defined over the label universe domain $\boldsymbol{\mathcal{Y}}$ and skeleton in $\#^{(i,c)}$ will be denoted as $\boldsymbol{\mathcal{Y}}^{\#^{(i,c)}}$. The void DAG will be denoted by the special symbol ξ.

In Figure 1, we have reported examples of visual patterns represented as DOAGs. Each

[1] If no supersource is present, a new node connected with all the nodes of the graph with null *indegree* can be added.

node of the graph represents a different colored component of the visual pattern, while edges are used to encode the concept of "connected with". The direction of the edges is decided on the basis of a procedure which scans the picture bottom up and left to right (see section 4.1 for more details). Labels are not shown in the figure.

3 Self-Organizing Maps for Structured Data

The aim of the Self-Organizing Map learning algorithm is to learn a *feature map* $\mathcal{M} : \boldsymbol{\mathcal{I}} \to \boldsymbol{\mathcal{A}}$ which given a vector in the spatially continuous input space $\boldsymbol{\mathcal{I}}$ returns a point in the spatially *discrete* output display space $\boldsymbol{\mathcal{A}}$. This is obtained in the SOM by associating each point in $\boldsymbol{\mathcal{A}}$ with a different neuron. Moreover, the display space $\boldsymbol{\mathcal{A}}$ topology is typically endowed by arranging this set of neurons as the computation nodes of a one- or two-dimensional lattice. Given an input vector $\boldsymbol{v}$, the SOM returns the coordinates within $\boldsymbol{\mathcal{A}}$ of the neuron with the closest weight vector. Thus, the set of neurons induces a partition of the input space $\boldsymbol{\mathcal{I}}$. In typical applications $\boldsymbol{\mathcal{I}} \equiv I\!R^m$, where $m \gg 2$, and $\boldsymbol{\mathcal{A}}$ is given by a two dimensional lattice of neurons. With this setting, high dimensional input vectors are projected into the two dimensional coordinates of the lattice, with the aim of preserving, as much as possible, the topological relationships among the input vectors. The SOM is thus performing data reduction via a vector quantization approach. When the input space is a structured domain with labels in $\boldsymbol{\mathcal{Y}}$, i.e., $\boldsymbol{\mathcal{I}} \equiv \boldsymbol{\mathcal{Y}}^{\#^{(i,c)}}$, the map $\widehat{\mathcal{M}} : \boldsymbol{\mathcal{Y}}^{\#^{(i,c)}} \to \boldsymbol{\mathcal{A}}$ can be realized by the following recursive definition:

$$\widehat{\mathcal{M}}(\mathcal{D}) = \begin{cases} nil_{\boldsymbol{\mathcal{A}}} & \text{if } \mathcal{D} = \xi \\ \mathring{\mathcal{M}}\left(\boldsymbol{y}_s, \widehat{\mathcal{M}}(\mathcal{D}^{(1)}), \ldots, \widehat{\mathcal{M}}(\mathcal{D}^{(c)})\right) & \text{otherwise} \end{cases}$$

where $nil_{\boldsymbol{\mathcal{A}}}$ is a special coordinate (the void coordinate) into the discrete output space $\boldsymbol{\mathcal{A}}$, and $\mathring{\mathcal{M}}$: $\boldsymbol{\mathcal{Y}} \times \boldsymbol{\mathcal{A}}^c \to \boldsymbol{\mathcal{A}}$ is a standard SOM, defined on a generic node, which takes in as input the label of the node and the "encoding" of the subgraphs $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(c)}$ according to the $\widehat{\mathcal{M}}$ map. By "unfolding" the recursive definition, it turns out that $\widehat{\mathcal{M}}(\mathcal{D})$ can be computed by applying $\mathring{\mathcal{M}}$ to leaf nodes first (i.e., nodes with zero outdegree), and proceeding with the application of $\mathring{\mathcal{M}}$ bottom-up from the frontier to the supersource of the graph $\mathcal{D}$.

Assuming that each label in $\boldsymbol{\mathcal{Y}}$ is encoded in $\boldsymbol{\mathcal{U}} \subset I\!R^m$, for each node v in vert($\mathcal{D}$), we have a vector $\boldsymbol{u}_v$ of dimension m. Moreover, the display output space $\boldsymbol{\mathcal{A}}$, where $\boldsymbol{\mathcal{A}} \equiv [1..n_1] \times [1..n_2] \times \cdots \times [1..n_q]$, is realized through a q dimensional lattice of neurons. Thus, the winning neuron is represented by the coordinates $(i_1, .., i_q)$. With the above assumptions, we have that $\mathring{\mathcal{M}}$: $I\!R^m \times ([1..n_1] \times \cdots \times [1..n_q])^c \to [1..n_1] \times \cdots \times [1..n_q]$, and the $m + cq$ dimensional input vector $\boldsymbol{v}$ to $\mathring{\mathcal{M}}$, representing the information about a generic node v, is defined as $\boldsymbol{v} = \left[\boldsymbol{u}_v \; \mathcal{D}_{ch_1[v]} \; \mathcal{D}_{ch_2[v]} \; \cdots \; \mathcal{D}_{ch_c[v]}\right]$, where $\mathcal{D}_{ch_i[v]}$ is the coordinates vector[2] of the winning neuron for the graph pointed by the i-th pointer of v.

Of course, each neuron j with coordinates vector $\boldsymbol{c}_j$ in the q dimensional lattice has an associated vector weight $\boldsymbol{w}_{\boldsymbol{c}_j} \in I\!R^{m+cq}$. The weights associated with each neuron in the q dimensional lattice $\mathring{\mathcal{M}}$ can be trained using the standard SOM's learning process, while the training algorithm for $\widehat{\mathcal{M}}$ is shown in Figure 2. For convenience we will refer to this algorithm as the SOM-SD (Self Organizing Map for Structured Data).

[2] The void coordinate $nil_{\boldsymbol{\mathcal{A}}}$ can be defined, for example, by the vector with all components at -1.

Unsupervised Stochastic Training Algorithm for $\widehat{\mathcal{M}}$

input: Set of training DAGs $T = \{\mathcal{D}_i\}_{i=1,\ldots,N}$, c maximum outdegree, map $\overset{\circ}{\mathcal{M}}$;

begin

randomly set the weights for $\overset{\circ}{\mathcal{M}}$;

repeat

randomly select $\mathcal{D} \in T$ with uniform distribution;

`List(`$\mathcal{D}$`)` $\leftarrow$ an inverted topological order for vert($\mathcal{D}$);

for $v \leftarrow$`first(List(`$\mathcal{D}$`))` **to** `last(List(`$\mathcal{D}$`))` **do**

train($\overset{\circ}{\mathcal{M}} \left(\left[\boldsymbol{u}_v \, \mathcal{D}_{ch_1[v]} \, \mathcal{D}_{ch_2[v]} \, \cdots \, \mathcal{D}_{ch_c[v]} \right] \right)$);

$\mathcal{D}_v \leftarrow \overset{\circ}{\mathcal{M}} \left(\left[\boldsymbol{u}_v \, \mathcal{D}_{ch_1[v]} \, \mathcal{D}_{ch_2[v]} \, \cdots \, \mathcal{D}_{ch_c[v]} \right] \right)$;

end

Figure 2: The unsupervised SOM-SD algorithm.

The coordinates for the (sub)graphs are stored in $\mathcal{D}_v$, once for each processing of graph $\mathcal{D}$, and then used when needed[3] for the training of $\overset{\circ}{\mathcal{M}}$. Of course, the stored vector is an approximation of the true coordinate vector for the graph rooted in v, however, if the learning rate η is small, this approximation may be negligible. The call $train()$ refers to a single step of standard SOM training.

The major innovation introduced with this algorithm is to treat each leaf node as a standard input for a self organising map, and the coordinates of the winning neuron as pointer within a parent node. In other words, we assume that the information transferred between the children and the parent is via the coordinates of the winning neuron in the child node.

4 Experiments

Artificial data sets obtained from a benchmark problem as described in [3] were generated and used for the experiments shown in this paper. The data sets consist of images specifically designed to evaluate adaptive data structure processing.

4.1 The data set

The data set was created using attributed plex grammars [2] for generating connected visual patterns. Sets from three different domains, ‘Traffic Policemen’, ‘Houses’, and ‘Sailing Ships’, were generated. Each domain was represented by $2,500$ images that were split up into two distinct equally sized data sets. When producing these sets care has been taken to obtain connected objects. The *traffic policeman* patterns were generated in compliance to the benchmark problem described in [3]. The final data sets are built by merging the three data sets.

Data structures are extracted from the images by using a simple scan line approach which extracts regions of pixels of the same color. The graph is generated representing each region

[3]Notice that the use of an inverted topological order guarantees that the updating of the coordinate vectors $\mathcal{D}_v$ is done before the use of $\mathcal{D}_v$ for training.

by a node and spatial relationships between regions by edges. Each node receives a two dimensional label stating the x,y coordinate of the center of gravity of the represented region.

Applied to the data patterns described previously, this algorithm generates graphs with the following properties:

	Outdegr.	Depth		Num. nodes	
Data set	Max.	Min	Max	Min	Max
Policemen	3	4	5	9	11
Houses	5	2	3	4	7
Ships	6	1	2	3	13

Hence, policemen patterns produce deep narrow graphs, ships and houses have a flat and wide data structure where some graphs are of identical structure (e.g. pattern `h2179` and pattern `s1609` as shown in figure 4 and figure 6). This points out that some distinguishing features of an image are encoded only in the labels.

As indicated earlier, these three sets are combined into one training set so that the maximum outdegree of all nodes in the training set is six. As a result, assuming $q = 2$, every input vector $\boldsymbol{v}$ is of dimension $2 + 2 \times 6$, where the first 2 is the dimension of the data label (m), and 6 the maximum outdegree (c). The $3,750$ graphs contained in the training set featured a total of $29,808$ nodes. The validation set which contained another $3,750$ graphs had a total of $29,984$ nodes.

4.2 Results

This section discusses the typical behaviour of a sample SOM network for data structures ($\mathcal{M}^{\#}$) when applied to the data set described earlier. Training is performed unsupervised. However, we utilize symbols to code the plots according to the original class membership of the data. The visualization of these classes gives implications on how well the network encodes structures and the data labels. All the results described in this section were obtained by a SOM network which uses a hexagonal topology and the Gaussian neighborhood function.

A SOM map with 12036 neurons arranged in a two-dimensional grid of size 102 horizontal and 118 vertical has been trained for 300 iterations (epochs). The outcome is displayed in Figure 3-*(a)*, which reports where leaf nodes (diamond shaped symbols), intermediate nodes (squares), and root nodes (pluses) are mapped in the lattice. It can be seen that these three instances of nodes are mapped in distinct areas with very limited overlap. In fact, there are relatively large areas containing codebook vectors that have not been activated by any of the nodes from the data set. These areas separate clusters of neurons formed by different types of nodes, and by different 'classes'. Figure 3-*(b)* shows a zoom into the location of neurons that were activated by root nodes only. Root nodes belonging to the domain 'House' are represented as squares, those associated with the domain 'Ship' are presented as crosses. The root nodes from the domain 'Policemen' are represented by diamond shaped symbols. It can be seen that the domains 'Houses', 'Ships', and 'Policemen' are well represented in individual, not overlapping clusters. Hence, SOM-SD was able to encode structural information well. Furthermore, it is found that for each domain cluster there are a number of sub-clusters. A collection of 30 sub-clusters have been named in Figure 3-*(b)*.

Figures 4, 5, and 6 present a zoom into the clusters formed by houses, policemen, and ships. A typical sample has been retrieved for many sub-clusters and are displayed accordingly. The graph representation of a sample is displayed to its right, and its i.d. (identity) is

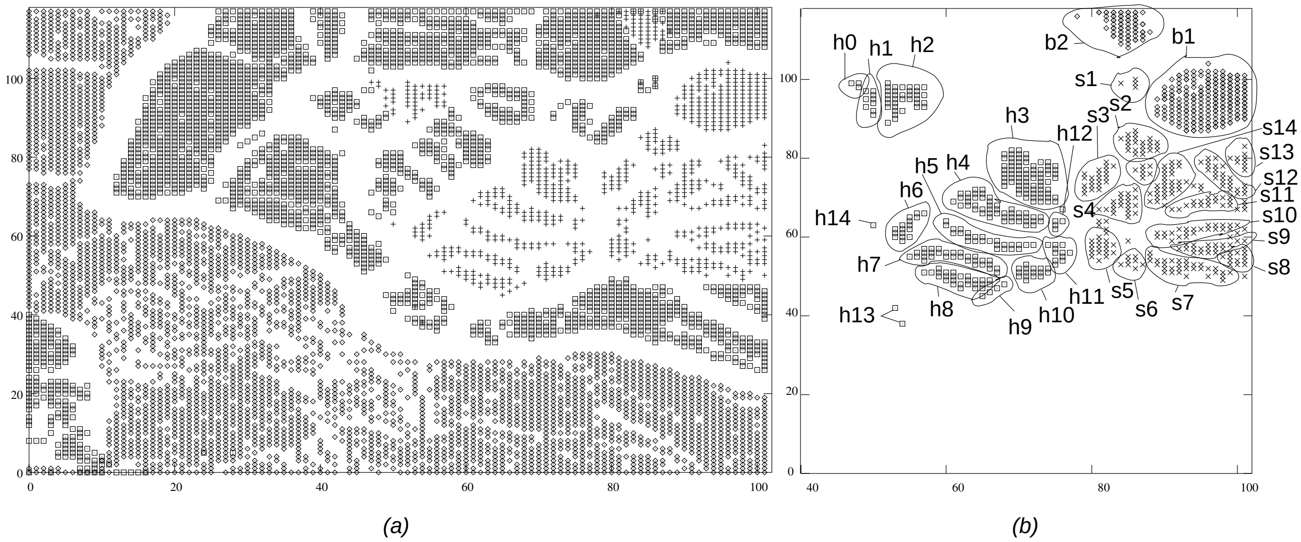

Figure 3: Clustering of training nodes in a SOM-SD map of size 102 × 118. *(a)*: location of all nodes from the training set; the location of root nodes are represented as pluses, intermediate nodes are given as squares, and leaf nodes are mapped in areas showing diamond shaped symbols. *(b)*: a zoom into the area on which root nodes are mapped reveals further details; 30 named sub-clusters are illustrated (houses are represented by squares, ships by crosses, and policemen by diamond shaped symbols).

plotted underneath. The i.d. identifies a pattern through a unique name. For example, the i.d. `h573` is associated with the 573rd house in the data set. To avoid further cluttering, the graph presentations do not show the data labels associated with its nodes.

When considering Figure 4, it can be seen that the map has performed feature mapping as one would expect; structures that are mapped within a sub-cluster share the same features, and structures mapped in nearby sub-clusters share similar features. In addition, it has been found that there is an order even within each sub-cluster. For example, we found that houses mapped onto cluster *h5* generally had a small roof when mapped closer to the left end of the cluster. Houses mapped closer to the right end of cluster *h5* featured a large roof. This agrees with the observation that structures in cluster *h6* which is located to the right of cluster *h5* typically show houses with a small roof, whereas houses with a large roof are found in clusters located to the right of *h5* (cluster `h10` and `h11`).

Some of the sample structures displayed in Figure 4 feature identical graph representations. Nevertheless, they are mapped onto different sub-clusters. A closer look at the structures named `h1179`, `h377`, `h1549`, and `h2263` is facilitated by displaying the structural representation and its data labels at the right side of Figure 4. Dictated by the scan-line algorithm employed to produce these graph structures, the root node is a representation of the houses' body, the data label associated is the center of gravity of the body. When looking from the left to right, the first offspring is the door, the next two offsprings represent the windows, and the last offspring of the root node is the roof. The node representing the roof has an offspring by its own. This is the chimney of the house. As can be seen, the Euclidean distance between the labels associated with nodes representing the windows is considerable larger than the difference between labels associated with the root node, and the nodes representing the door and roof. The Euclidean distance between labels associated with the chimney is also large. We also found that patterns located in the same sub-cluster showed windows located in the same configuration. For example, almost all structures mapped in the same sub-cluster

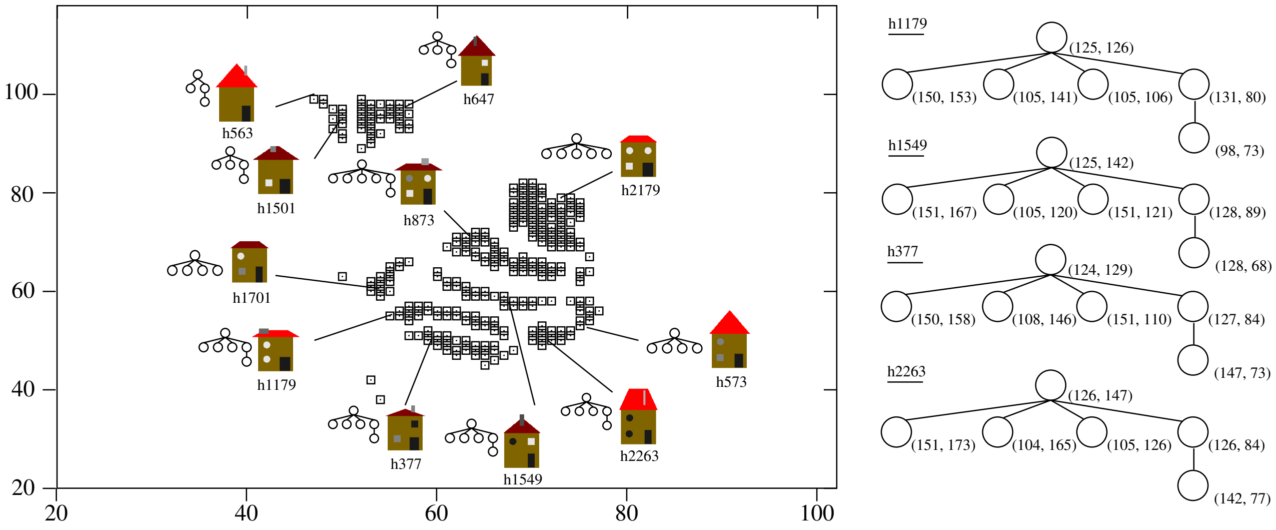

Figure 4: Zoom into the clusters formed by root nodes belonging to the class 'House'.

as the pattern named `h1549` showed two windows in the upper row. This observation leads to the conclusion that structures may be mapped in distinct sub-clusters in accordance to information stored in the data label, where data labels located closer to the root node have a stronger effect on where a pattern is eventually mapped.

The clusters involving the policemen are shown in Figure 5. Apparently, the policemen were mapped only into two distinct sub-clusters. We found that the policemen produced only two different graph structures depending on whether the original image featured a policemen with long pants where the legs were not visible, or with both legs visible. The former produced graphs where the root node (the pedestal) had only one offspring (the pants), whereas in the other case the root node had two offsprings (the two legs). These two structures formed the two distinct clusters in the map. Any other information, such as the location of the arm was encoded in the data label. When looking closer at the mapping of the policemen within each sub-cluster, we found that there is a mapping according to information present in the data label. For example, the cluster formed by images showing policemen with both legs visible, mostly featured images showing a policemen with a raised left arm if mapped closer to the upper right corner of that cluster. Patterns mapped closer to the lower right corner had both arms lowered, those mapped nearer to the lower left area showed a raised left arm, and patterns with both arms raised were typically mapped near the upper left corner.

A zoom into sub-clusters formed by the instance 'Ship' is displayed in Figure 6. Ships could feature as many as three windows, and up to three masts which were with or without sails, or with or without flags. This class of patterns had the largest range of differently structured graphs. This is reflected in a relatively large number of sub-clusters which are not always clearly separated. Nevertheless, we found that patterns within a sub-cluster shared similar features, and patterns mapped onto nearby sub-clusters shared more similarities than those mapped in distant sub-clusters. However, the mapping is not as 'clean' as for the instance *houses*. In particular, features that were located deeper inside the graph structure (which represented flags and sails) have not been mapped in distinct areas. We believe that the map has not been chosen large enough to allow a distinct mapping of such a large number of different instances, so that the network is forced to focus on more relevant information

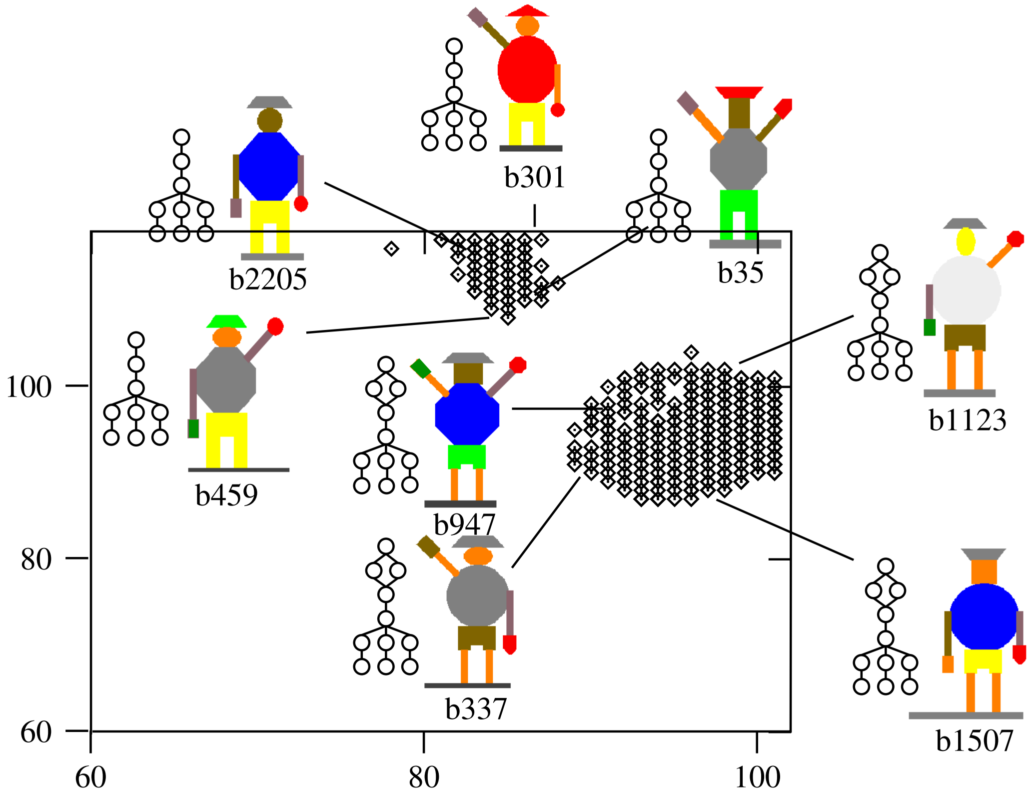

Figure 5: Zoom into the clusters formed by root nodes belonging to the class 'Policeman'.

presented closer to the root node.

The results displayed in Figures 4, 5, and 6 have demonstrated that the SOM-SD training algorithm is well able to encode both structural information as well as data presented in the data label. The quality of the mapping can be adjusted by adopting a weighted Euclidean metric for the input vectors, where the information on the label is weighted in a different way with respect to the information on the pointers. This allows SOM-SD to focus more strongly on the mapping of structural details or on the data labels.

When testing the network with the validation data, we obtained exactly the same results we have just described for the training data. This is mainly due to the fact that the training set was highly representative of the data statistics.

The task performed by the proposed network, i.e., the development of a "topographic" map of the input (components of) structures in which the spatial locations of the neurons are indicative of statistical features of the (components of) structures, is to the best of our knowledge new, especially when considering labelled DAGs where the labels are real-valued vectors. Here we stress the differences of our approach with respect to both standard SOMs and similarity measures defined for graphs.

Although SOMs have been defined to work on vector spaces, a few examples of extensions to sequences exist. The best known works are certainly the Temporal Kohonen Map (TKM) [1], and the Recurrent Self-Organizing Map (RSOM) [10]. Both models use leaky integrators for the representation of the state (context). A different approach is adopted in the

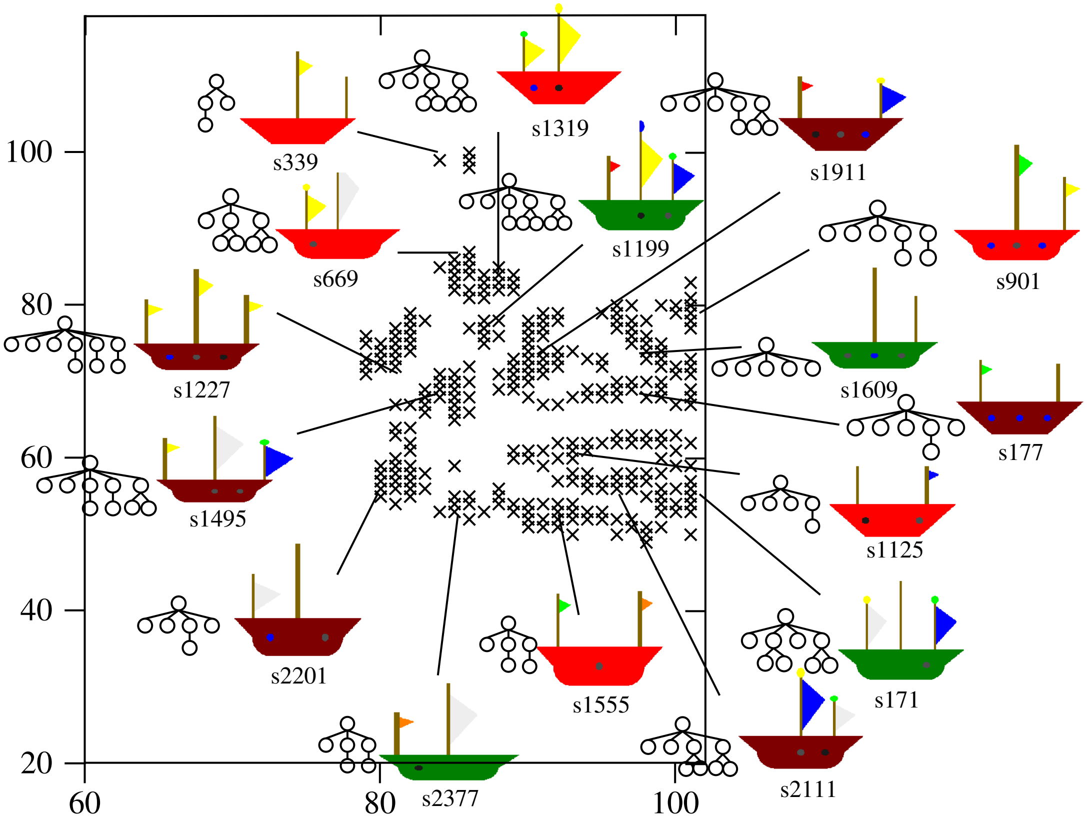

Figure 6: Zoom into the clusters formed by root nodes belonging to the class 'Ships'.

Contextual Self-Organizing Map (CSOM) [11], where recurrent connections between units in the map are used. We are not aware of any SOM model able to deal with more general data structures. Moreover, our computation of context is far more efficient and simple than the ones used in any of the above cited models.

Concerning similarity measures defined for structured objects, we have to distinguish between measures that may potentially be used for building up a topographic map and actual systems able to perform at least clustering of structures. Let us first of all consider the *graph edit distance*. The basic idea is to define the distance between two structured objects as the (weighted) number of basic edit operations which must be performed to transform one structure into the other. There are at least two problems with the graph edit distance: *i)* edit costs should be defined a priori; *ii)* computational complexity is high.

Concerning systems, few of them are able to cluster (and not to build up a "topographic" map) discrete-valued, structural databases: we are aware of Labyrinth [8] and SUBDUE [4]. Our model is similar in many respects to SUBDUE, e.g., (approximate) discovery of substructures that best compress the input graphs, with the difference that we have real-valued labels, and a "topographic" map is built. However, in our case we are only able to deal with DAGs rather than with more general data structures at present.

5 Conclusions

In this paper, we proposed a new way of addressing unsupervised learning problems involving structured objects (not just processed visual patterns) using a self organising map technique. Specifically, we showed how a SOM network can be used recursively in order to learn a map defined on DAGs.

We demonstrated the capabilities of the proposed model by utilizing a relatively large data set taken from an artificial benchmark problem involving visual patterns encoded as labelled DAGs. The experimental results clearly show that the proposed model is able to encode structural features in a coherent way. The information on the labels, on the other side, can be encoded finely only if the map is large enough.

Anyway, it is clear that our model is in fact able to build a two dimensional metric on the structured input space. This ability of the model opens the doors to a set of new application domains, involving data mining and knowledge discovery, which were not possible to directly attack in the past for real-valued, structured objects.

As a final remark on the computational efficiency of the proposed model, it must be stressed out that recent techniques based on random projections [5], could be used in the future to speed up the selection of the winning neuron.

References

[1] G. Chappell and J. Taylor. The temporal kohonen map. *Neural Networks*, 6:441–445, 1993.

[2] T. Feder. Plex languages. *Info. Sciences 3*, pages 225–241, 1971.

[3] Markus Hagenbuchner and Ah Chung Tsoi. The traffic policeman benchmark. In Michel Verleysen, editor, *European Symposium on Artificial Neural Networks*, ISBN 2-9600049-9-X, pages 63–68. D-Facto, April 1999.

[4] L. Holder and D. Cook. Discovery of inexact concepts from structural data. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):992–994, 1993.

[5] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. 30th Symposium on Theory of Computing (STOC'98)*, pages 604–613, 1998.

[6] Teuvo Kohonen. *Self-Organisation and Associative Memory*. Springer, 3rd edition, 1990.

[7] A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–735, 1997.

[8] K. Thompson and P. Langley. *Concept Formation: Knowledge and Experience in Unsupervised Learning*, chapter Concept formation in structured domains, pages 127–161. Morgan Kaufmann Publishers, 1991.

[9] Marc van Hulle, editor. *Faithful Representations and Topographic Maps*. New York ; John Wiley, 2000.

[10] M. Varsta, J. Del, R. Millan, and J. Heikkonen. A recurrent self-organizing map for temporal sequence processing. In *Proc. 7th Int. Conf. Artificial Neural Networks, ICANN'97*, pages 421–426, 1997.

[11] T. Voegtlin. Context quantization and contextual self-organizing maps. In *Proc. Int. Joint Conf. on Neural Networks*, volume VI, pages 20–25, 2000.